UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/789,311	02/27/2004	Sheueling Chang Shantz	6000-31500	9201

58467 7590 11/28/2007

MHKKG/SUN
P.O. BOX 398
AUSTIN, TX 78767

EXAMINER	
JOHNSON, CARLTON	
ART UNIT	**PAPER NUMBER**
2136	

MAIL DATE	DELIVERY MODE
11/28/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/789,311	SHANTZ ET AL.	
	Examiner	Art Unit	
	Carlton V. Johnson	2136	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 September 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-67* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-67* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responding to application papers filed on **2-27-2004**.

2. Claims **1 - 67** are pending. Claims **1, 4, 5, 6, 10, 11, 12, 13, 18, 19, 21, 22, 23, 24, 29, 30, 31, 32, 33, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67** have been amended. Claims **1, 21, 38, 53, 66, 67** are independent.

Response to Arguments

3. Applicant's arguments filed 9/19/2007 have been fully considered but they are not persuasive.

3.1 Applicant argues that the referenced prior art does not disclose, *"the first partial result representing the high order bits summed with low order bits of a result of a first number multiplied by a second number". (see Remarks Page 18)*

The claim limitations merely recite arithmetic operations which are performed on integer values. The Gressel and Stribaek prior art combination discloses arithmetic operations performed on integer values. The stated types of operations indicated by the prior art discloses

The Gressel prior art discloses the results of a first arithmetic operation used as input to another arithmetic operation. A bit value of an arbitrary length is a partial result. The high order bits are a partial result. The low order bits are a partial result. The two partial results are combined by adding the products. This is equivalent to a fist partial

result representing the high order bits summed with the low order bits of a result of a first number multiplied by a second number:

3.2 Applicant argues that the referenced prior art does not disclose, *"Applicants assert that Stribaek does not overcome the deficiencies of Gressel in teaching the specific limitations of claim 1". (see Remarks Page 19)*

The Stribaek prior art is not used to overcome any deficiencies in the Gressel prior art. There are no deficiencies in the Gressel prior art. The claims limitations the Stribaek prior art used to reject are clearly stated in the Office Action.

3.3 The referenced prior art clearly discloses the generation of a partial result, the usage of that partial result in subsequent arithmetic operations. The referenced prior art clearly show the complete set of the types of arithmetic operations disclosed in the claims limitations (XOR operation, multiplication, carry add operations, carry save operations) The Office Action indicates a citations for each independent and each dependent claim rejection. *(see Remarks Page 17-20)*

3.4 The Gressel prior art discloses arithmetic operations such as multiplication and addition utilizing the partial results of the first operation. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

In very long instruction word (VLIW) architectures, which include many microcode

architectures, multiple simultaneous operations and operands are specified in a single instruction. (http://www.answers.com/topic/instruction-computer-science) This standard computer architecture feature discloses a single arithmetic instruction to perform multiple operations.

An instruction also designates the destination address (memory locations, registers) for the results of the completion of an instruction. (*"On traditional architectures, an instruction includes an opcode specifying the operation to be performed, such as "add contents of memory to register", and zero or more operand specifiers, which may specify registers, memory locations, or literal data ": http://www.answers.com/topic/instruction-computer-science*)

The Gressel prior art discloses partial results from an arithmetic operation. The partial results could be the high order bits. (see Gressel col. 2, lines 31-37: arithemtic operations utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

The Gressel prior art discloses storing and utilizing the results of an operation in subsequent arithmetic operations. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into a next (subsequent) operation)

3.5 The examiner has considered the applicant's remarks concerning In response to executing an arithmetic instruction, a first number is multiplied by a second number, and a partial result from a previously executed single arithmetic instruction is fed back from a first carry save adder structure generating high order bits of the current arithmetic

instruction to a second carry save adder tree structure being utilized to generate low order bits of the current arithmetic instruction to generate a result that represents the first number multiplied by the second number summed with the high order bits from the previously executed arithmetic instruction. Execution of the arithmetic instruction may instead generate a result that represents the first number multiplied by the second number summed with the partial result and also summed with a third number, the third number being fed to the carry save adder tree structure. Applicant's arguments have thus been fully analyzed and considered but they are not persuasive.

After an additional analysis of the applicant's invention, remarks, and a search of the available prior art, it was determined that the current set of prior art consisting of Gressel (6,748,410) and Stribaek (7,181,484) discloses the applicant's invention including disclosures in Remarks dated September 19, 2007.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims **1 - 67** are rejected under 35 U.S.C. 101 because the claimed invention is based on non-statutory subject matter and directed towards nothing more than the abstract idea of a mathematical algorithm. Abstract ideas are not eligible for patent protection. A claimed invention reciting a computer program product that solely calculates a mathematical formula or a computer readable medium that solely stores a mathematical formula is not directed to the type of subject matter eligible for patent

protection.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless -
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

7. Claims **1 - 10, 12 - 19, 21 - 29, 32 - 36, 38 - 46, 48 - 60, 62 - 67** are rejected under 35 U.S.C. 102(e) as being anticipated by **Gressel et al.** (US Patent No. **6,748,410**).

Regarding Claim 1, Gressel discloses a method implemented in a device supporting a public-key cryptography application, the method comprising: a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed arithmetic instruction in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of arithmetic structures; and the second arithmetic circuit, generating a first partial result of a currently executing arithmetic instruction in the public-key cryptography application, the first partial result representing the high order bits summed with low order bits of a result of a first number multiplied by a second number, the

summing of the high order bits being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit; storing the first partial result; and using the stored first partial result in a subsequent computation in the public-key cryptography application. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations; col. 29, lines 43-49: redundant representation of numbers; col. 1, lines 39-45; col. 5, lines 23-25: acceleration, improvements of arithmetic operations; col. 3, lines 28-32: arithmetic operations utilized to generate cryptographic key(s); col. 3, lines 18-22: processor utilization for key generation)

Regarding Claim 2, Gressel discloses the method as recited in claim 1 wherein the high order bits are fed back in redundant number representation. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 2, lines 31-37: multiplication two values, summing two values

utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 29, lines 43-49: redundant representation of numbers)

Regarding Claim 3, Gressel discloses the method as recited in claim 2 wherein the redundant number representation includes sum and carry bits. (see Gressel col. 29, lines 43-49: redundant representation of numbers; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder)

Regarding Claim 4, Gressel discloses the method as recited in claim 1 further comprising feeding back the high order bits through a register to the second arithmetic circuit. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 5, Gressel discloses the method as recited in claim 1, further comprising: generating a second partial result of the currently executing arithmetic instruction in the first arithmetic circuit, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 6, Gressel discloses the method as recited in claim 1 further comprising: generating a second partial result of the currently executing arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number summed with the high order bits of the previously executed arithmetic instruction. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 7, Gressel discloses the method as recited in claim 6 further comprising supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from

previous multiplication)

Regarding Claim 8, Gressel discloses the method as recited in claim 5 wherein the generating of the first and second partial result is in response to execution of a single arithmetic instruction. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 9, Gressel discloses the method as recited in claim 6 wherein the generating of the first and second partial result is in response to execution of a single arithmetic instruction. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 10, Gressel discloses the method as recited in claim 1 wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32;

col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 12, Gressel discloses the method as recited in claim 1 wherein at least one of the first and second pluralities of arithmetic structures is usable to perform both integer and XOR multiplication. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 13, Gressel discloses the method as recited in claim 12, further comprising a logical circuit in at least one of the first and second arithmetic circuits supplying a fixed value if in XOR multiplication mode or a variable value for integer multiplication mode that varies according to inputs supplied to the logical circuit if in integer multiplication mode, to thereby ensure a result is determined in XOR multiplication unaffected by carry logic performing carries in integer multiplication mode. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-

out)

Regarding Claim 14, Gressel discloses the method as recited in claim 13 wherein the logical circuit operates as a majority circuit in integer multiplication mode and outputs a zero in the XOR multiplication mode. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 15, Gressel discloses the method as recited in claim 1 wherein the first partial result is in redundant number representation. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 29, lines 43-49: redundant representation of numbers)

Regarding Claim 16, Gressel discloses the method as recited in claim 15 further comprising supplying the first partial result to an adder circuit to generate a non redundant representation of the first partial result and a carry out value. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 29, lines 43-49: redundant representation of numbers; col. 31, lines 46-

48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 17, Gressel discloses the method as recited in claim 16 further comprising feeding back the carry out value to the adder circuit. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 18, Gressel discloses the method as recited in claim 16, further comprising feeding back the carry out value to the second arithmetic circuit. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 19, Gressel discloses the method as recited in claim 1, further comprising feeding back high order bits of the currently executing arithmetic instruction from the first arithmetic circuit to the second arithmetic circuit for use with execution of a subsequent single arithmetic instruction. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 3, lines 1-7; col. 53,

lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation)

Regarding Claim 21, Gressel discloses a method implemented in a device supporting public-key cryptography application, the method comprising: a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed arithmetic instruction in the public-key cryptography application, generated by the first arithmetic circuit to a second arithmetic circuit comprising a second plurality of arithmetic structures; supplying a third number to the second arithmetic circuit; the second arithmetic circuit generating a first partial result of a currently executing arithmetic instruction in the public-key cryptography application, the first partial result being a representation of the high order bits summed with, low order bits of a result of a first number multiplied by a second number, and with the third number, the summing being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit; storing the first partial result; and using the first partial result in a subsequent computation in the public-key cryptography application. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48;

col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations; col. 29, lines 43-49: redundant representation of numbers; col. 1, lines 39-45; col. 5, lines 23-25: acceleration, improvements of arithmetic operations; col. 3, lines 28-32: arithmetic operations utilized to generate cryptographic key(s); col. 3, lines 18-22: processor utilization for key generation)

Regarding Claim 22, Gressel discloses the method as recited in claim 21 further comprising feeding back the high order bits through a register to the second arithmetic circuit. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 23, Gressel discloses the method as recited in claim 21, further comprising: the first arithmetic circuit generating a second partial result of the currently executing arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49:

arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 24, Gressel discloses the method as recited in claim 21 further comprising: generating a second partial result of the currently executing arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number summed with the high order bits of the previously executed arithmetic instruction and the third number. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 25, Gressel discloses the method as recited in claim 24 further comprising supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-

49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 26, Gressel discloses the method as recited in claim 23 wherein the generating of the first and second partial result is in response to execution of a single arithmetic instruction. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 27, Gressel discloses the method as recited in claim 21 supplying the first partial result to an adder circuit to generate a non redundant representation of the first partial result and a carry out value. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines

47-51: carry-out)

Regarding Claim 28, Gressel discloses the method as recited in claim 27 further comprising feeding back the carry out value to the adder circuit. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 29, Gressel discloses the method as recited in claim 27, method further comprising feeding back the carry out value to the second arithmetic structures. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 32, Gressel discloses the method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures is usable to perform both integer and XOR multiplication. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 33, Gressel discloses the method as recited in claim 32 further comprising a logic circuit in at least one of the first and second pluralities of arithmetic structures supplying a fixed value if in XOR multiplication mode or a variable value that varies according to inputs supplied to the logical circuit if in integer multiplication mode, to thereby ensure a result is determined in XOR multiplication unaffected by carry logic performing carries in integer multiplication mode. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 34, Gressel discloses the method as recited in claim 33 wherein the logic circuit operates as a majority circuit in integer multiplication mode and outputs a zero in the XOR multiplication mode. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 35, Gressel discloses the method as recited in claim 21 wherein the high order bits are in redundant number representation. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 29, lines 43-49: redundant representation of numbers)

Regarding Claim 36, Gressel discloses the method as recited in claim 21 further comprising feeding back high order bits of the currently executing arithmetic instruction from the first arithmetic circuit to the second arithmetic circuit for use with execution of a subsequent single arithmetic instruction. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation)

Regarding Claim 38, Gressel discloses a processor configured to support public-key cryptography applications, comprising: a first plurality of arithmetic structures configured to generate high order bits for an arithmetic operation in a public-key cryptography application that includes a multiplication operation; and a second plurality of arithmetic structures configured to generate low order bits of the arithmetic operation; wherein the second arithmetic structures are further configured to receive the high order bits generated by the first plurality of arithmetic structures during a previous arithmetic

operation in the public-key cryptography application and to generate a first partial result of the arithmetic operation, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation; and wherein the processor further comprises a register configured to stopre the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations; col. 29, lines 43-49: redundant representation of numbers; col. 1, lines 39-45; col. 5, lines 23-25: acceleration, improvements of arithmetic operations; col. 3, lines 28-32: arithmetic operations utilized to generate cryptographic key(s); col. 3, lines 18-22: processor utilization for key generation)

Regarding Claim 39, Gressel discloses the processor as recited in claim 38, wherein the first arithmetic structures are configured to generate a second partial result of the arithmetic instruction, the second partial result representing the high order bits of the

arithmetic operation. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 40, Gressel discloses the processor as recited in claim 39, wherein the second arithmetic structures are further configured to supply values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 41, Gressel discloses the processor as recited in claim 39, wherein the first and second arithmetic structures are configured to generate of the first and second partial results in response to execution of a single arithmetic instruction. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation

or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder)

Regarding Claim 42, Gressel discloses the processor as recited in claim 38, further comprising a register coupled to the first and second arithmetic structures to supply the high order bits to the second arithmetic structures. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 43, Gressel discloses the processor as recited in claim 38, wherein the first partial result is in redundant number representation. (see Gressel col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 29, lines 43-49: redundant representation of numbers)

Regarding Claim 44, Gressel discloses the processor as recited in claim 43, further comprising an adder circuit configured to receive the first partial result and to generate a

non redundant representation of the first partial result and a carry out value. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 45, Gressel discloses the processor as recited in claim 44, wherein adder circuit is configured to feed the carry out value back to itself as an input. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 46, Gressel discloses the processor as recited in claim 44, wherein adder circuit is configured to feed the carry out value back to the second arithmetic structures. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 48, Gressel discloses the processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 49, Gressel discloses the processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures is configured to selectively perform one of integer and XOR multiplication according to a control signal. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 50, Gressel discloses the processor as recited in claim 49, further comprising a plurality of logic circuits in the first and second pluralities of arithmetic structures, each logic circuit responsive to the control signal to supply a fixed output value in XOR multiplication mode and a variable output value in integer multiplication

mode, the variable output value varying according to values of inputs supplied to the logic circuit, to thereby ensure a result is determined in XOR multiplication mode unaffected by carry logic generating carries in integer multiplication mode. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 51, Gressel discloses the processor as recited in claim 50, wherein the logical circuit is configured to operate as a majority circuit in integer multiplication mode and to output a zero in XOR multiplication mode. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 52, Gressel discloses the processor as recited in claim 38, wherein the processor is a general purpose processor. (see Gressel col. 3, lines 18-22:

processor utilization for key generation)

Regarding Claim 53, Gressel discloses a processor configured to support public-key cryptography applications, comprising: a first plurality of arithmetic structures configured to generate high order bits for an arithmetic operation in a public-key cryptography application that includes a multiplication operation of a first and a second number; a second plurality of arithmetic structures configured to generate low order bits of the arithmetic operation; wherein the second arithmetic structures are configured to: receive the high order bits generated by the first plurality of arithmetic structures during a previous arithmetic operation; receive a third number; and generate a first partial result of the arithmetic operation, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation, and with the third number; and wherein the processor further comprises a register configured to store the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67;

col. 41, lines 20-23: register usage; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations; col. 29, lines 43-49: redundant representation of numbers; col. 1, lines 39-45; col. 5, lines 23-25: acceleration, improvements of arithmetic operations; col. 3, lines 28-32: arithmetic operations utilized to generate cryptographic key(s); col. 3, lines 18-22: processor utilization for key generation)

Regarding Claim 54, Gressel discloses the processor as recited in claim 53, wherein the first arithmetic structures are further configured to generate a second partial result of the arithmetic instruction, the second partial result representing the high order bits of the arithmetic operation. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 55, Gressel discloses the processor as recited in claim 54, wherein the second arithmetic structures are further configured to generate values in one or more most significant columns and to supply them to one or more least significant columns of the first arithmetic structures while generating the first and second partial results. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col.

11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 56, Gressel discloses the processor as recited in claim 54, wherein the first arithmetic structures are configured to generate of the first and second partial result in response to execution of a single arithmetic instruction. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication)

Regarding Claim 57, Gressel discloses the processor as recited in claim 53, further comprising a register coupled to the first and second arithmetic structures to supply the high order bits to the second arithmetic structures. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 58, Gressel discloses the processor as recited in claim 53, further comprising an adder circuit configured to receive the first partial result and to generate a non redundant representation of the first partial result and a carry out value. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 59, Gressel discloses the processor as recited in claim 58, wherein the adder circuit is further configured to feed the carry out value back to itself as an input. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage)

Regarding Claim 60, Gressel discloses the processor as recited in claim 58, method

wherein the adder circuit is further configured to feed the carry out value back to the second arithmetic structures. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 62, Gressel discloses the processor as recited in claim 53, wherein at least one of the first and second arithmetic structures comprises carry save adder tree columns. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out)

Regarding Claim 63, Gressel discloses the processor as recited in claim 53, wherein the arithmetic structures are configured to selectively perform one of integer and XOR multiplication according to a control signal. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order

bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 64, Gressel discloses the processor as recited in claim 63, further comprising a plurality of logic circuits in at least one of the first and second pluralities of arithmetic structures, each logic circuit responsive to the control signal to supply a fixed output value in XOR multiplication mode and a variable output value in integer multiplication mode, the variable output value varying according to values of inputs supplied to the logic circuit, to thereby ensure a result is determined in XOR multiplication mode unaffected by carry logic generating carries in integer multiplication mode. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 65, Gressel discloses the processor as recited in claim 64, wherein the logical circuit is configured to operate as a majority circuit in integer multiplication mode and to output a zero in the XOR multiplication mode. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit

operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations)

Regarding Claim 66, Gressel discloses an apparatus configured to support a public-key cryptography application comprising: means for feeding back high order bits of a previously executed arithmetic instruction, generated by a first arithmetic circuit, to a second arithmetic circuit generating low order bits of a currently executing arithmetic instruction; means for using the second arithmetic circuit to generate a first partial result of the currently executing arithmetic instruction, the first partial result representing the high order bits of the previously executed arithmetic instruction that are summed with low order bits of a multiplication result of a first number multiplied by a second number; means for using the first partial result in a subsequent computation in the public-key cryptography application. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage; col. 8, lines 59-60; col. 53, lines 13-19: XOR operations; col. 29, lines 43-49: redundant representation of numbers; col. 1, lines 39-45; col. 5, lines 23-25:

acceleration, improvements of arithmetic operations; col. 3, lines 28-32: arithmetic operations utilized to generate cryptographic key(s); col. 3, lines 18-22: processor utilization for key generation)

Regarding Claim 67, Gressel discloses an apparatus configured to support a public-key cryptography application comprising: means for feeding back high order bits of a previously executed arithmetic instruction, from a first arithmetic circuit that generated the high order bits, to a second arithmetic circuit generating low order bits of a currently executing arithmetic instruction; means for supplying a third number to the second arithmetic circuit; and means for using the second arithmetic circuit to generate a first partial result, the first partial result being a representation of the high order bits of the previously executed arithmetic instruction summed with low order bits of a result of a first number multiplied by a second number and with the third number; and means for using the first partial result in a subsequent computation in the public-key cryptography application. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure; col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication; col. 6, lines 20-25: adder; col. 31, lines 46-48; col. 6, line 66 - col. 7, line 9; col. 31, lines 44-46: carry-save adder; col. 49, lines 47-51: carry-out; col. 2, lines 4-9; col. 5, lines 58-67; col. 41, lines 20-23: register usage; col.

8, lines 59-60; col. 53, lines 13-19: XOR operations; col. 29, lines 43-49: redundant representation of numbers; col. 1, lines 39-45; col. 5, lines 23-25: acceleration, improvements of arithmetic operations; col. 3, lines 28-32: arithmetic operations utilized to generate cryptographic key(s); col. 3, lines 18-22: processor utilization for key generation)

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims **11, 20, 30, 31, 37, 47, 61** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Gressel et al.** (US Patent No. **6,748,410**) in view of **Stribaek et al.** (US Patent No. **7,181,484**).

Regarding Claim 11, Gressel discloses the method as recited in claim 1 wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of Wallace tree columns. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure) Gressel does not specifically disclose whereby a plurality of Wallace tree columns. However, Stribaek discloses wherein further comprises a plurality of Wallace tree columns. (see Stribaek col. 9, lines 10-24; col. 9, lines 37-39:

Wallace tree)

It would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of Wallace tree multiplication. One of ordinary skill in the art would have been motivated to employ the teachings of Stribaek in order to enable the capability for extended precision in arithmetic calculations due to extensive and increasing usage of public key cryptography. (see Stribaek col. 1, lines 61-67: " *... Public-key cryptosystems have been used extensively for user authentication and secure key exchange, while private-key cryptography has been used extensively to encrypt communication channels. As the use of public-key cryptosystems increases, it becomes desirable to increase the performance of extended-precision modular arithmetic calculations. ...* ")

Regarding Claim 20, Gressel discloses the method as recited claim 1 further comprising storing the high order bits. (see Gressel col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication) Gressel does not specifically disclose whereby an extended carry register. However, Stribaek discloses wherein an extended carry register. (see Stribaek col. 9, lines 10-24; col. 9, lines 37-39: Wallace tree; col. 5, lines 41-45: extended carry operations)

It would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of extended carryoperations. One of ordinary skill in the art would have been motivated to employ the teachings of

Stribaek in order to enable the capability for extended precision in arithmetic calculations due to extensive and increasing usage of public key cryptography. (see Stribaek col. 1, lines 61-67)

Regarding Claim 30, Gressel discloses the method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure) Gressel does not specifically disclose a plurality of Wallace tree columns. However, Stribaek discloses wherein further comprises a plurality of Wallace tree columns. (see Stribaek col. 9, lines 10-24; col. 9, lines 37-39: Wallace tree; col. 2, line 66 - col. 3, line 6: public key cryptographic calculations)

It would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of Wallace tree multiplication. One of ordinary skill in the art would have been motivated to employ the teachings of Stribaek in order to enable the capability for extended precision in arithmetic calculations due to extensive and increasing usage of public key cryptography. (see Stribaek col. 1, lines 61-67)

Regarding Claim 31, Gressel discloses the method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures. Gressel does not specifically disclose carry save adder tree columns. However, Stribaek discloses

wherein further comprises a plurality of carry save adder tree columns. (see Stribaek col. 9, lines 10-24; col. 9, lines 37-39: Wallace tree; col. 5, lines 41-45: extended carry operations; col. 7, lines 31-37; col. 9, lines 10-14: carry-save adder; col. 2, line 66 - col. 3, line 6: public key cryptographic calculations)

It would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of carry save adder and Wallace tree multiplication. One of ordinary skill in the art would have been motivated to employ the teachings of Stribaek in order to enable the capability for extended precision in arithmetic calculations due to extensive and increasing usage of public key cryptography. (see Stribaek col. 1, lines 61-67)

Regarding Claim 37, Gressel discloses the method as recited in claim 21 further comprising storing the high order bits. (see Gressel col. 2, lines 31-37: multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication) Gressel does not specifically disclose whereby an extended carry register. However, Stribaek discloses wherein an extended carry register. (see Stribaek col. 9, lines 10-24; col. 9, lines 37-39: Wallace tree; col. 5, lines 41-45: extended carry operations)

It would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of extended carry operations (register). One of ordinary skill in the art would have been motivated to employ the teachings of Stribaek in order to enable the capability for extended precision in

arithmetic calculations due to extensive and increasing usage of public key cryptography. (see Stribaek col. 1, lines 61-67)

Regarding Claim 47, Gressel discloses the processor as recited in claim 38, wherein at least one of the first and second pluralities of the arithmetic structures. (see Gressel col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5: arithmetic structure) Gressel does not specifically disclose whereby a plurality of Wallace tree columns. However, Stribaek discloses wherein further comprises a plurality of Wallace tree columns. (see Stribaek col. 9, lines 10-24; col. 9, lines 37-39: Wallace tree)

It would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of Wallace tree multiplication. One of ordinary skill in the art would have been motivated to employ the teachings of Stribaek in order to enable the capability for extended precision in arithmetic calculations due to extensive and increasing usage of public key cryptography. (see Stribaek col. 1, lines 61-67)

Regarding Claim 61, Gressel discloses the processor as recited in claim 53, wherein at least one of the first and second arithmetic structures comprises Wallace tree columns. (see Gressel col. 3, lines 1-7; col. 53, lines 13-19; col. 53, lines 49-51: feedback of a previous operation into next operation; col. 3, lines 28-32; col. 11, lines 7-11; col. 11, lines 40-49: arithmetic operation or instructions; col. 31, lines 44-46; col. 41, lines 3-5:

arithmetic structure) Gressel does not specifically disclose whereby further comprising Wallace tree columns. However, Stribaek discloses wherein further comprises a Wallace tree column. (see Stribaek col. 9, lines 10-24; col. 9, lines 37-39: Wallace tree)

It would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of Wallace tree multiplication. One of ordinary skill in the art would have been motivated to employ the teachings of Stribaek in order to enable the capability for extended precision in arithmetic calculations due to extensive and increasing usage of public key cryptography. (see Stribaek col. 1, lines 61-67)

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later

than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Carlton V. Johnson whose telephone number is 571-270-1032. The examiner can normally be reached on Monday thru Friday , 8:00 - 5:00PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Nasser Moazzami can be reached on 571-272-4195. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

C.J.

CVJ
November 13, 2007

NASSER MOAZZAMI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

11/26/07

Carlton V. Johnson
Examiner
Art Unit 2136